COMSC-210 Lab 7
Stacks and Queues

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this assignment you will develop your own implementations of stacks and queues. You will have to decide upon design considerations and any additional functionality for yourself.

After you complete this lab assignment, post the required files to the COMSC server so that you receive the proper credit for this 50-point lab. Your posted files are due at midnight of the evening of the due date indicated in the course outline. Use the "Submit your FA2015 work" link on the class website to post your file for this lab, using the **lab7** folder provided.

LAB 7a: Write And Apply A Stack Template `[ Stack.h, StackDriver.cpp, and Calculator7.cpp ]`
Purpose. In this lab you will write your own stack template, using either array or linked-list implementation, per your choice.

Requirements. Rewrite lab 2c's `Calculator2.cpp` as **`Calculator7.cpp`**, but use your own stack template. Name your class `Stack`, and write its specification and function templates in a file named **`Stack.h`**. Use either a dynamic array or a linked list to implement the class -- your choice.

In addition to the calculator application, and prior to developing it, write a driver program, **`StackDriver.cpp`**, to *fully* test your templated stack class. Refer to the lecture topic 3 notes for how to test a class, which includes all functions, ifndef, object copy, and object assignment testing.

You decide on the "design consideration" issues that are listed in the lecture topic 7 notes. Decide on the pop, push, and peek options -- you do not have to choose the same as is indicated in the notes. If you use a linked list, decide if you want a header node or not. But do not use either the `StaticArray`, `DynamicArray`, or `LinkedArray` template that you developed, and do not use any STL containers in your `Stack`, and do *not* include any `cout` statements in the template.

Submit the *three* files (2 CPPs and 1 H) to the class website for credit.

Program I/O. Same as for `Calculator2.cpp`.

LAB 7b: Write And Test A Queue Template `[ Queue.h and QueueDriver.cpp ]`
Purpose. In this lab you will write your own queue template, using either array or linked-list implementation, per your choice.

Requirements. Rewrite your own templated queue class. Name your class `Queue`, and write its specification and function templates in a file named **`Queue.h`**. Use either a dynamic array or a linked list to implement the class, and be sure to include dynamic memory management.

Write a driver program, **`QueueDriver.cpp`**, to *fully* test your templated stack class.

You decide on the "design consideration" issues that are listed in the lecture topic 7 notes. Decide on the pop, push, and peek options -- you do not have to choose the same as is indicated in the notes. If you use a linked list, decide if you want a header node or not. But do not use either the `StaticArray`, `DynamicArray`, or `LinkedArray` template that you developed, and do not use any STL containers in your `Stack`, and do *not* include any `cout` statements in the template.

Submit the two files (1 CPP and 1 H) to the class website for credit.

Program I/O. All console I/O, in the driver program only.

LAB 7c: Use A Stack `[ PreProcessor.cpp ]`
Purpose. In this lab you use a stack to solve the "matching brackets" problem. Your program will read an input text file, and report the first-found container mismatches, such as unbalanced parentheses.

Requirements. Write **`PreProcessor.cpp`**, that prompts the user for an input filename, and checks it for matching container brackets. The input file should be expected to be a text file, of type H, CPP, JAVA, HTML, or JS. Check for these bracket combinations:

- parentheses `(` and `)`
- curly braces `{` and `}`
- square brackets `[` and `]`
- quotes `"` and `"` -- the same symbols
- comment offsets `/*` and `*/` -- in a CPP, H, JAVA, or JS file

Just report the first found occurrence of a mismatch.

Once a `/*` is found, IGNORE everything until its matching `*/` is found. Once a `//` is found, IGNORE everything else on the same line. Once a `"` is found, IGNORE everything else on the same line until another `"` is found. If none is found, end with mismatch found.

Do NOT check for angle brackets, because there can be less-than and greater-than symbols that make that hard to manage.

Use a stack data structure as the basis of your solution, but *you* decide whether to use your own `Stack` template that you developed for lab 7a, or the STL `stack`.

Note that when using a `string` for a filename, the "open" statement in `ifstream` is supposed to use a C-string as a parameter, and not a C++ `string`. Your compiler may overlook this non-standard behavior, but your instructor will not. [Read more...]

Fully test. Every one of the H files you've created for this class should run successfully through your preprocessor, so test with those. The PreProcessor.cpp file itself should fail -- test with it, too, and verify that it finds the first error where you think if should. Submit the completed CPP file to the class website for credit.

Program I/O. Input: using `cin`, a filename. Include the expected filename extensions in the prompt (H, CPP, JAVA, HTML, or JS) for reference only -- that is, don't validate. Output: If there are no mismatches, say so in your own words. Otherwise, output the line number of the line containing the opening bracket, say what that bracket is, and say that you cannot find the closing backet and what that bracket should be. Or, output the line number of the line containing the closing bracket for which there is no opening bracket, say what that bracket is, and say that you cannot find the opening backet and what that bracket should be.

Example (computer prompts in bold).

Example: file with no mismatches
Enter filename (H, CPP, JAVA, HTML, or JS): x.cpp
No mismatches found by my preprocessor in x.cpp

Example: file with no closing parenthesis
Enter filename (H, CPP, JAVA, HTML, or JS): x.cpp
Opening parenthesis found in line 10 of x.cpp
But no matching closing parenthesis found

Example: file with no opening square bracket
Enter filename (H, CPP, JAVA, HTML, or JS): x.cpp
Closing square bracket found in line 10 of x.cpp
But no matching opening square bracket found

How to pause a console program: *show / hide*

GRADING RUBRIC *show/hide*
